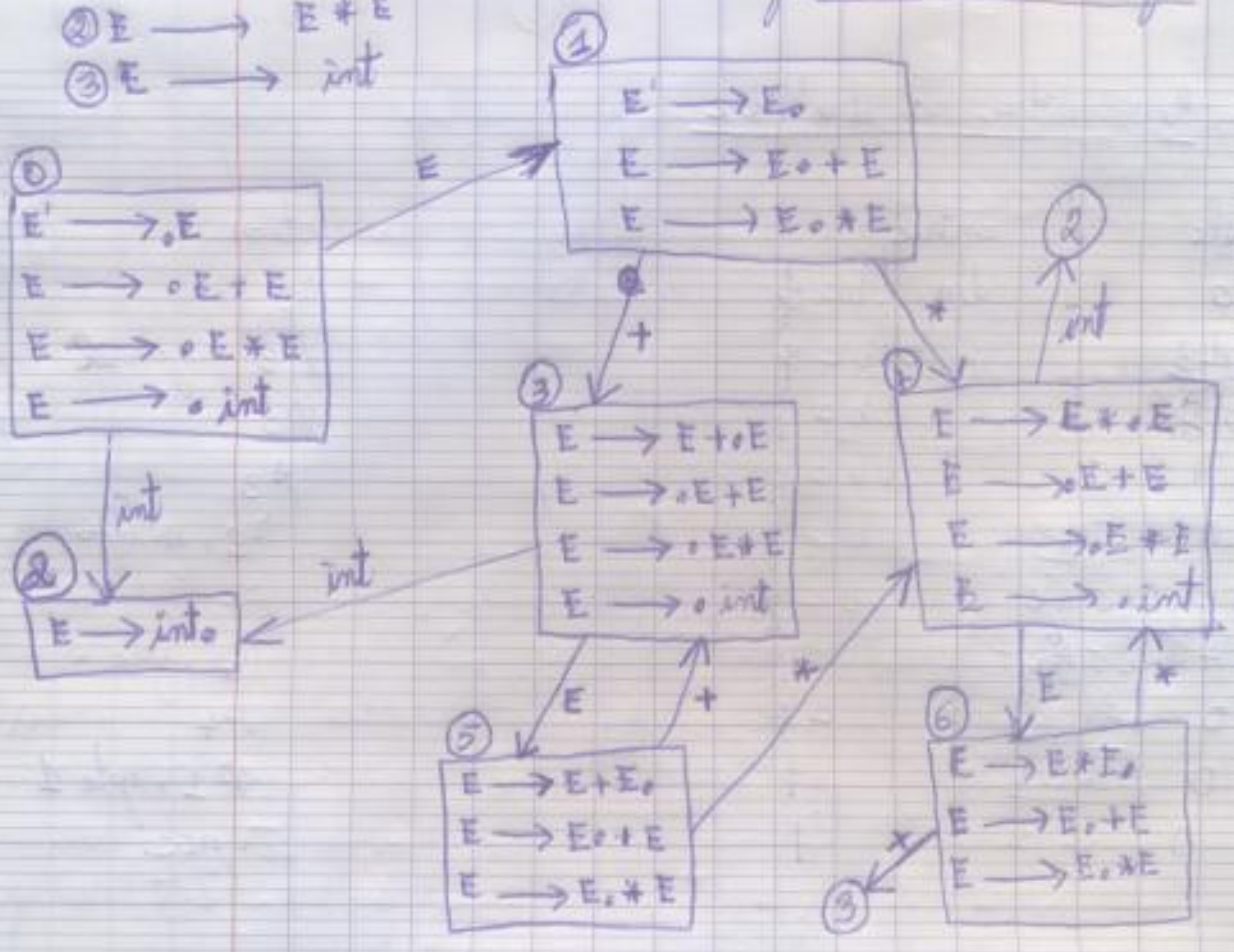
0 E' → E
① E → E + E
② E → E * E
③ E → int

→ grammaire ambigue



premier(E) = { int }
premier(E') = { int }.

suivant(E') = { $ }.
suivant(E) = { $, +, * }

	shift / reduce				goto
	int	+	*	$	E
0	Δ2				1
1		Δ3	Δ4	acc	
2		r3	r3	r3	
3	Δ2				5
4	Δ2				6
5		r1 Δ3	r1 Δ4	r1	
6		r2 Δ3	r2 Δ4	r2	

Δi ≡ shift i
ri ≡ reduce i